PRACTICAL PROGRAMMING IN BASIC
76811Y
Ed 1

BASIC IS AFTER ALL, BASIC

76811Y/8311

Printed in South Africa

ICS
SCHOOL OF COMPUTER TRAINING

PROGRAMMING IN BASIC

BASIC Is After All, BASIC

BASIC Is After All, BASIC

SIMILARITIES AND DIFFERENCES BETWEEN VARIOUS VERSIONS OF BASIC

In an ideal world, every computer language would have a standard set of rules and conventions for its use. In a few cases (notably COBOL, FORTRAN and PL/I), such rules do exist. Representatives of industry and government meet periodically in Washington, D.C. at the American National Standards Institute (ANSI) to review these widely used languages. The Institute sets forth certain rules and standards as to how these languages will be used.

Once this body has approved language standards, all computer manufacturers and software developers must follow the agreed-upon rules. Thus, for example, a programmer who knows ANSI COBOL can code a program which can run (with little or no modifications) on virtually any system.

FIGURE 1 — Within almost any group of people speaking the same language, you can discover a multitude of regional dialects, unique ways of expressing thoughts, and variations in word meanings. The same phenomenon has occurred among writers of BASIC: they all write the same language, but subtle differences creep into their finished work. Computer manufacturers are also widening language differences by structuring ROM and RAM in unique ways.

Many other languages do not have this compatibility because of their more limited usage. But why has BASIC, now the most widely known of all computer languages, not adopted a uniform structure?

The answer lies in the history of the computer revolution and the evolution of BASIC. It was originally developed as a simple means of teaching programming to people who were not scientists, mathematicians or engineers (who mainly used FORTRAN or machine code). Its popularity rapidly spread throughout schools and universities across the country. But as long as computers were located only in large companies and colleges, most professional software was not written in BASIC.

With the explosion of the fourth generation of computers, small but powerful computing power became available to more and more people. Microcomputers are purchasable by the smallest of businesses and on the most limited family budget.

Thus, microcomputer manufacturers such as ATARI, APPLE, COMMODORE, RADIO SHACK, TIMEX SINCLAIR and others, chose BASIC as the language which would appeal most to their enthusiastic but inexperienced new clients.

Unfortunately, as competition became keener, they developed their hardware and software independently and secretly. This has left us with a multitude of "BASICS" such as MICROSOFT BASIC, APPLE BASIC, TRS-80 BASIC, C-BASIC, and M-BASIC, to name but a few. Each has its own peculiarities, but these are far outweighed by their similarities to each other. It is the intent of this supplement to ease you through this maze so that you can feel comfortable working with any of them.

Hope is on the horizon for standards lovers. The ANSI Committee now has a standard, though limited, vocabulary of BASIC keywords. This may prove to be the start of "ANSI BASIC"!

MAKING THE CONVERSION

Once you have a working knowledge of one of the versions of BASIC, you are well on your way to understanding them all. Here are a few simple ways to convert to a new version:

1. *The Manuals*

   A manual is supplied with the computer or diskette which contains the BASIC interpreter. You will have to familiarize yourself with it to learn how to enter, edit, run and save your programs. These commands must be mastered before you do any coding. For example, on some computers, the BASIC interpreter resides in ROM and when the system is "booted-up", you can immediately begin entering your program. However, on others, BASIC must be loaded first from tape, diskette or disk.

   Editing and executing your program will also differ. Your manual will explain these procedures – hopefully with some examples!

FIGURE 2 – Explaining to others how to program on various computers is a major business. This firm does it via book and mail order.

(From Popular Computing T.M.)

2. *Reference and Textbooks*
For many of the more widely used microcomputers, independent authors have developed books purporting to teach BASIC for a specific machine. If you find your manual difficult to understand (or incomplete), one of these types of books might prove to be a good substitute.

3. *Trial and Error*
This method may be too frustrating an experience for the insecure, but may be a good way for the daring programmer to start hands-on coding quickly on a new system. Before describing this technique, a word of caution is in order: Read the operating manual first!

Although you probably won't do any damage by depressing the wrong key, an hour or so of careful reading may save hours of powering off and powering on the system as one repeatedly gets into a hopeless confusion of error messages and "frozen" keyboards. In more extreme cases, users have been known to accidently "erase" the BASIC interpreter or the operating system from tape or disk. Your hour or two of reading will certainly be well spent. Once you are familiar with your system, you can begin using this "trial-and-error method".

A. Begin by taking a copy of a small program which you have written, tested and debugged on your first system.

B. Enter your old program onto the new system, line by line. Because of the interpretive nature of BASIC, each statement will be checked for syntax errors.

C. If your statement is accepted as written, then you have probably used a compatible BASIC keyword in both of the systems.

D. If your statement is not accepted, then you have discovered a dissimilarity between them.

E. If the syntax error (because of the syntax cursor position or error message) points to the *keyword* as the culprit, then you will have to search through the manual of the new system to find the one or more keywords which can be substituted. If your reference source has a summarized listing of keywords (most do), you may be freed from a more detailed examination of the manual.

F. If the syntax error references the entries made *after* the keyword, then you're in luck! The keyword is the same, but the two formats are different. Your manual will certainly have a definition of the format of each of the keywords; the problem may be in the sequence of your entries or the punctuation used or the functions on the line. In any case, you probably won't have to change very much.

G. Once you have your entire program entered, you can then run it. Of course, you might assume that RUN is universally used. This is *almost* true! (Some computers may use the single-letter command, "R", to execute.)

H. If your output is good, then you are well on your way to mastering the new version. If not, you have a little more research to do. The difference is a little bit more difficult to detect, since it is a "logic" error. (First test your original, though. Maybe it didn't work on the first system either!)

Whatever method you employ to learn your new version of BASIC, you must learn to read the manual eventually. That is why

If you're ready to learn intermediate BASIC programming

STEP BY STEP TWO IS READY FOR YOU!

In Each Parentheses, Choose One:

If you're one of the (happy, smart, ecstatic) thousands who learned (quickly, clearly, non-boringly) to use their APPLE computers with our beginners' BASIC tutorial, *The New Step By Step*, then we (know, believe, swear) that *Step By Step Two* is ready to (ease, breeze, squeeze) you into the (advanced, intermediate, grown-up) world of PEEK and POKE, hexidecimal numbers, concatenations, and (much, much, much) more.

On the other (hand, foot) if you didn't (grow with, thrill to, involve the whole family in) the PDI (learning, hands-on, fun) experience, here's what you missed:

- Teaching techniques that teach
- Involvement you enjoy
- Sounds that spur you on
- Graphics that simplify the complex
- Animation that makes this tutorial a stimulating experience.
- A friendly voice that guides you to course completion

The *Step By Step Two* program works this way:

- the computer program sets up screen displays or sample programs for you.
- the cassette voice tells you what's happening.
- you (deal with, figure out, guess at) the answer.
- the computer (praises, pans) your work.
- you (peruse, plunge into, practice in) the Work Book.
- after each lesson, you're (quizzed, queried, questioned).
- you're then (prepared, practiced, primed) for the next lesson.
- the final exam reveals all (superstars, slackers).

There's lots to learn in *Step By Step Two:*

- PEEK & POKE
- Default values
- Memory map
- CALL program
- ASCII codes
- CHR$ and ASC functions
- control characters
- RAM vs ROM
- hexidecimal numbers
- machine monitor
- string logic
- string arrays
- high resolution graphics
- screen memory

But don't take our (word, words, wordiness) about how (good, great, grand) the *Step By Step* method is. Listen to our (critics, reviewers, friends):

"If you want to learn BASIC or would like a little guidance and encouragement added to what you already know, then the way to go is *Step By Step.*" —Softalk

"The *Step By Step* approach is the next best thing to having an experienced programmer by your side... *Step By Step* is a superb example of a successful blend of various media. The teaching principles are sound, the execution is virtually flawless, and the whole thing works." —Popular Computing.

If you want to move ahead in BASIC programming, the next (simple, logical, shrewd) step is *Step By Step Two.*

Step By Step Two is available at fine retail stores or direct from PDI for $89.95 plus $3.00 shipping and handling. (The package includes back-up discs.)

P.S. If you've yet to take your first step into BASIC, it's time to get *The New Step By Step*! Same great tutorial techniques, for $79.95 plus $3.00 shipping and handling.

PDI

Program Design, Inc.
11 Idar Court
Greenwich, CT 06830
203-661-8799

*Apple is a trademark of Apple Computer Corp.

FIGURE 3 — Programmed tutorials are showing up as computer companies strive to gain a stronger relationship with users.

(From Popular Computing T.M.)

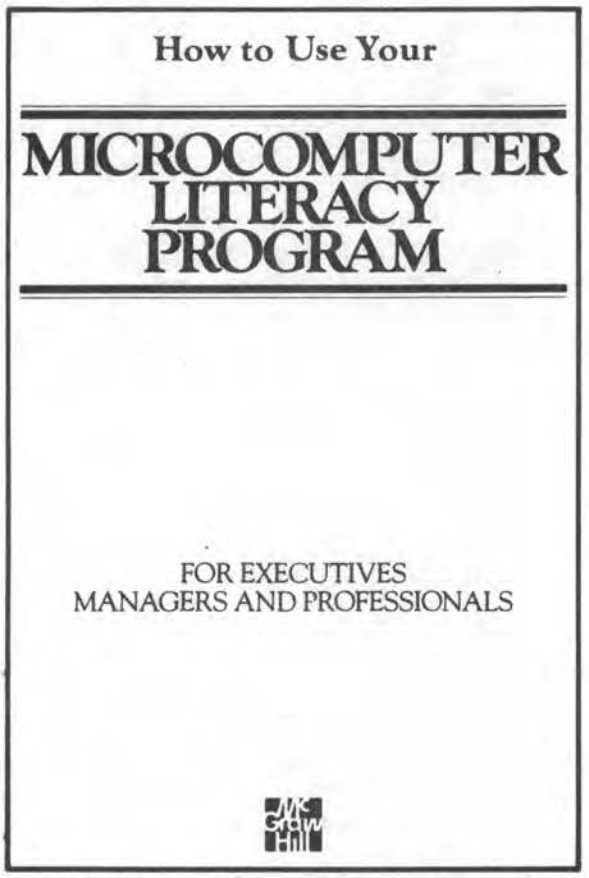


FIGURE 4 – McGraw Hill has developed a combination audiotape and text program for managers and executives. The program addresses most issues facing managers – including the matter of understanding differences among BASIC dialects.

some people will choose software based upon the clarity of the accompanying text.

And don't stop once you can do everything on the new system that you did on the old one. Each version has its own extensions and nuances that give it capabilities you may wish to use.

GENERAL SIMILARITIES AND DIFFERENCES

All BASIC statements consist of a line number, a keyword and operands. Sometimes multiple instructions can be placed on one line by separating them with a colon (:) or back slash (/); e.g.

```
1Ø  LET A = 5: LET B = 1
```

This option is generally not recommended for widespread use by any but the most experienced of programmers, as debugging, line insertion and line deletion become much more complicated!

VARIABLE NAME FORMATION

The rules for forming variable names are *not* universal. Some allow for only two characters for a numeric variable (e.g. T1, G, HT), while others allow more characters to give a more descriptive name (e.g. SALES, AMOUNT, TOTAL).

On many systems, the second (or last) character of the variable name determines its type. Thus, a dollar sign "$" may define a string variable, while a percent sign "%" defines a numeric integer.

Check your manual for this information.

ARITHMETIC

The "LET" statement is almost always used to assign a value to a variable, although on some computers, the use of the keyword may be omitted; e.g.

```
1Ø  LET A = 1
```

OR

```
1Ø  A = 1
```

The arithmetic operators (+, –, *, /) are widely used. Make sure that you are aware of the sequence in which complex arithmetic expressions will be calculated.

FUNCTIONS

Each version of BASIC has its own set of predefined mathematical functions. Some require parentheses be used to enclose the argument of the function (e.g. SQR (4) to get the square root of 4 or SQR 4).

BENTON HARBOR BASIC

and

EXTENDED BENTON HARBOR BASIC

This Appendix is a typical "User's Manual", as supplied to describe the features of a particular version of BASIC.

This BENTON HARBOR BASIC User's Manual illustrates the scope of the special instructions you can expect from such a user's Manual, and will further illustrate the features of BENTON HARBOR BASIC, which we have used for our course material.

FIGURE 5 – The understanding of a BASIC dialect is often taught in classes for computer operators and executives. This presentation of Benton Harbor BASIC is from a McGraw Hill program.

Once again, your manual ought to have a list and examples of the functions available to you.

BRANCHING AND LOOPING

The "GOTO" statement will almost always work, as will "GOSUB" and "RETURN" ("RET" is a commonly accepted abbreviation). Some computers allow for conditional branching to occur as a variation such as "ON . . . GOSUB . . . " or "ON . . . GOTO . . .".

These statements can be explained best by an example of their use:

```
6Ø  ON C GOTO 1ØØ, 15Ø, 2ØØ
```

When executed, this statement will cause a branch to line number 1ØØ if the numeric variable C has a value of 1; line 15Ø if it equals 2, and line 2ØØ for a value of 3. On some systems, you can substitute "GOSUB" and the program will access the different subroutines, once again depending upon the value of C. This statement has a limited applicability, however, since values other than 1, 2, or 3 in this example would be invalid. It is best used to cause branching depending on the response a user might make in selecting from a program menu. If your system doesn't have these options, your inconvenience will be slight. Merely use simple "IF" statements to control your logic flow; e.g.

```
6Ø  IF C = 1 THEN GOTO 1ØØ

7Ø  IF C = 2 THEN GOTO 15Ø

8Ø  IF C = 3 THEN GOTO 2ØØ
```

CONDITIONAL STATEMENTS

The "IF" keyword is another common one. The "IF . . . THEN" statement executes the instruction after the word "THEN", only if the statement is true.

In some BASICS, a variation known as the "IF . . . THEN . . . ELSE" can also cause unique processing to occur when the condition is false.

For example:

```
1Ø  IF A >  B THEN LET T = T + 1
    ELSE LET T = T – 1
```

This would increment T if the condition is true or decrement T if it is false. If you can't use the "ELSE" clause, an additional statement may be necessary, as in:

```
1Ø  IF A >  B THEN LET T = T + 1

2Ø  IF A <  = B THEN LET T = T – 1
```

BASE NUMBERING CONVENTIONS

Some systems start counting with 1 while others start with Ø. This can affect the

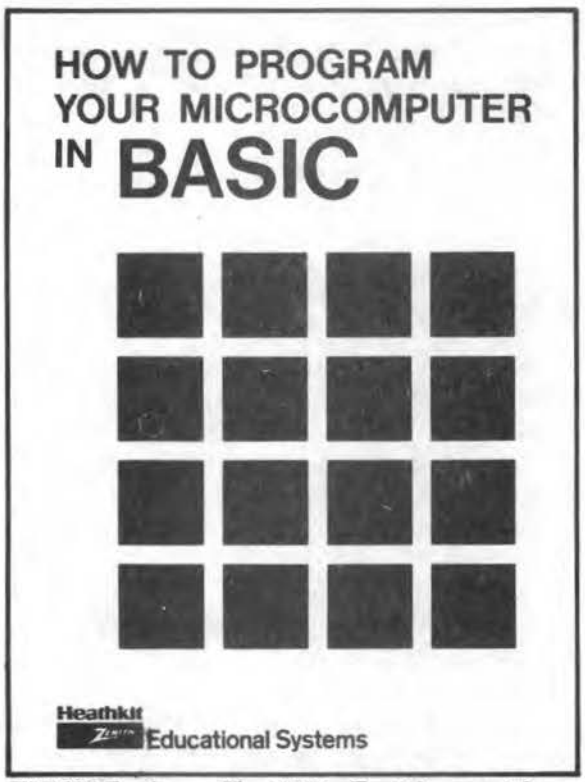


FIGURE 6 – Heathkit Zenith provides a comprehensive program on how to use BASIC with their computers. The program is in the form of several lessons with programming tasks.

printing of output and the dimensioning and accessing of tables. A statement such as:

```
1Ø  DIM T(1Ø)
```

may set up a table of 1Ø elements (Base 1) or a table of 11 elements (Base Ø). A quick check of the manual should resolve this.

(NOTE: On some systems the base number can be altered by another program statement as in: 1Ø OPTION BASE 1.)

INPUT STATEMENTS

Nowhere can the diversity of versions of BASIC be better seen than in the many ways that input and output statements are used.

Essentially there are three ways that data can be submitted as input to a BASIC program.

1. *READ . . . DATA*
   In this method, also known as internal data, the data is stored within the program in a DATA statement. This pool of values is assembled into an internal file such that a READ statement will substitute each successive value for a variable.

   Examine this program:

```
1Ø  READ A
2Ø  IF A = 999 THEN GOTO 6Ø
3Ø  PRINT A
4Ø  GOTO 1Ø
5Ø  DATA 2, 8, 6, 999
6Ø  STOP
```

   If this program were run, the READ statement on line 1Ø would use the first value in the DATA statement on line 5Ø (2) as the value of A and this would then be printed on the screen. The next loop would then use the next value (8) as the new value of A; the third loop would substitute 6.

   On the fourth loop, the value 999 would be accessed. This would cause the "IF" statement to cause a branch to the end of the program. This is a convenient way of ending loops in READ . . . DATA statements.

   The use of internal data is limited, however, since the data can only be changed by modifying the program. It is best used to store data of a relatively permanent nature, such as data for constants or tables.

2. *READ #n*
   On some systems, data can be stored

externally on tape or disk. In order to access this data within a program, the file must first be opened by name, as in:

```
1Ø  OPEN #1 : "NAME = SALES
    FILE"
```

The number given to the file is irrelevant initially. But when a READ statement is executed with the same number, a record will be input to the program, as in:

```
2Ø  READ #1 : N$, A
```

On many systems having this capability, it is necessary to describe the format of the input record (how long each variable is and its type) on a separate line known as a FORM line. For example:

```
1Ø  OPEN #1 : "NAME = SALES
    FILE"

2Ø  READ #1, USING 3Ø : N$, A

3Ø  FORM : C 15, N 5.2
```

This would access an external file called SALES FILE. Each occurrence of the READ would get values for two variables from each record. N$ would be the first 15 characters on each record. "A" would use the next five characters as numeric values with two decimal positions.

The manner in which the loop is terminated will depend on the format used in your system. Some use the EDF (End of File) clause to cause a branch when the last record has been read from the file.

This method allows for data to be shared by an unlimited number of programs while only needing to maintain it once.

3. *INPUT Statement*
The input statement allows data to be entered interactively and as such is the most common method of entering data. Some versions will let you input several variables from one INPUT statement, as in:

```
1Ø  INPUT A, C, T$
```

OUTPUT STATEMENTS

The means by which data is displayed or printed is probably the least standard of all BASIC operations. It would be beyond the scope of this supplement to describe all of the variations. There is really no substitute for your manual in solving this incompatability problem.

Here, however, are some common variations:

1. A simple PRINT instruction usually will cause output to be displayed on the next available line on the screen. If the screen is full, some programs will crash while others will automatically scroll the top line off the screen, making the last line available for the output.

2. The TAB clause is often used to position output on a particular column of the print line; e.g.

   ```
   1Ø  PRINT TAB 1Ø; "HELLO"
   ```

   will cause the message to be displayed at the 1Øth or 11th column (depending on the base numbering system in effect).

   Sometimes the number must be enclosed within parentheses, as in TAB(1Ø).

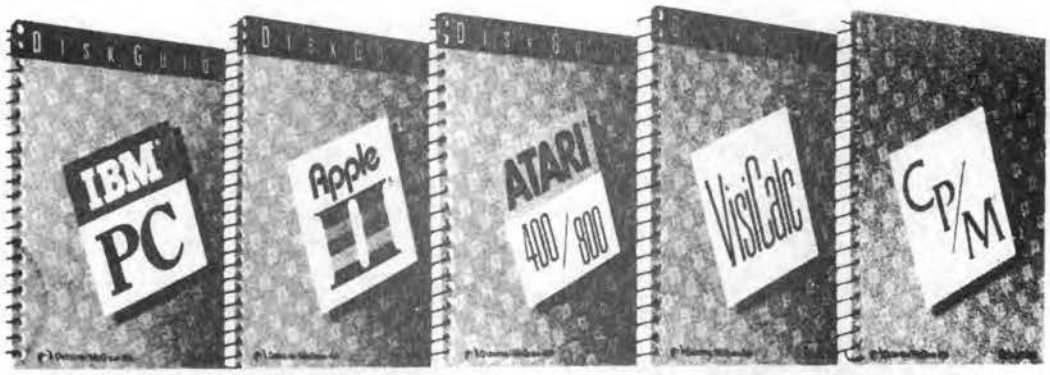
3. When two values are entered in a TAB statement, the first value references the line number and the second value, the column. Thus,

   ```
   1Ø  PRINT TAB(3, 16); "A"
   ```

   would print the character on line 3 and column 16.

Frustration Insurance.

The New DiskGuide™ Series from Osborne/McGraw-Hill.



- **IBM® PC DiskGuide** Order #94-2 $8.95
- **Apple® II DiskGuide** (Including Apple II Plus and IIe) Order #96-9 $7.95
- **ATARI® 400/800™ DiskGuide** Order #95-0 $7.95
- **VisiCalc® DiskGuide** Order #98-5 $6.95
- **CP/M® DiskGuide** (Including CP/M-80 and CP/M-86) Order #97-7 $8.95

The Computer Frustration Syndrome. It can hit anyone who uses a personal computer. We want to protect you from it. That's why we created **The DiskGuide Series.** Five easy-to-use, compact, computer reference guides chock full of vital commands, keys, summary tables and charts—information you need for frustration-free computing. No chitchat. Just clear-cut cues to help you master the Apple® II, IBM® PC, ATARI® 400/800™ plus VisiCalc® and CP/M.®

Look for **The DiskGuide Series** soon at a bookstore or computer store near you. Or order direct by phone or mail.

Escape The Computer Frustration Syndrome by reaching for our new **DiskGuide Series** or any one of 60 other Osborne/McGraw-Hill computer books.

OSBORNE/McGraw-Hill

TO ORDER:
By phone, call TOLL FREE 800-227-2895. In California, call 800-772-4077. VISA and MasterCard accepted.
By mail, complete the coupon below and mail to Osborne/McGraw-Hill, 2600 Tenth Street, Berkeley, CA 94710. All orders must be prepaid. You can pay by check, money order, VISA or MasterCard. Be sure to add taxes and shipping fees. See schedule below. Taxes and Shipping Fees. Add per item: $0.75 4th class, $1.50 UPS, $3.00 1st class/UPS Blue Label. California residents add local sales tax.
Allow 4-6 weeks for delivery. Prices subject to change without notice.

_____ YES, please send me a free catalog. Dept. B-2

Name _____

Address _____ City _____

State _____ Zip _____

Indicate method of payment:

_____ Check or money order _____ VISA/Exp. date _____

_____ MasterCard/Exp. date _____ Card # _____

Qty	Order #	Price

Tax _____

Shipping _____

Signature _____ Total _____

Osborne/McGraw Hill
2600 Tenth Street, Berkeley, CA 94710

FIGURE 7 – Making it easy to shift from one system to another has led major companies and entrepreneurs to produce cross-reference guides for various BASIC dialects and computer CPUs.

A simple variation of this is the "AT" clause:

```
1Ø  PRINT AT 3, 16; "A"
```

would accomplish the same output.

4. Formating the output is sometimes available with the "using" option. In this example, another line is referenced by the print statement which describes the way the output is to be displayed:

```
1Ø  PRINT, USING 2Ø : A
2Ø  FORM # # # . # #
```

This program would cause variable "A" to be displayed as a five-digit number with two decimal places (e.g. 15Ø.ØØ). The "PIC" clause is sometimes used on a FORM statement to give a "picture" of the way the data is to be displayed.

5. Just as data can be read from an external file in some BASIC versions, so it can also be entered on a file as output.

   The WRITE # n following an OPEN # n defining an output file usually accomplishes this.

6. Putting your output on a hard-copy printer uses common keywords such as LPRINT, COPY, and LLIST, but there are many variations; check your manual.

7. Punctuation is almost always critically important in a PRINT statement. Commas (,) usually cause printing in predefined zones, while semicolons (;) allow printing in the next available space. Often, these two operators cannot be used on the same PRINT instruction.

ONE FINAL WORD

As different as the many versions of BASIC may first appear, don't panic. In a very short time you will be able to switch from one to another comfortably. As you learn more about the new system, jot down good notes. These will help prevent making the same mistake twice.

Your first experience in BASIC will be the most difficult to master; each succeeding version will take less and less time to learn.

And remember, a good working knowledge of BASIC is usually acquired through extensive use of one or two systems at most. It is better to know one version well than to know only a little bit of many.